Ansible - Overview

- Ansible is an IT automation tool. It can configure systems, deploy software, and orchestrate more advanced IT tasks such as continuous deployments or zero downtime rolling updates.
- It's developed in python
- Easy to use & learn - because of yml syntex.
- Latest Version : 2.9
- Docs : https://docs.ansible.com/
- Why it's easy and popular: Ansible communicates with remote machines over the SSH protocol. By default, Ansible uses native OpenSSH and connects to remote machines using your current user name, just as SSH does.
- Typically you'll work with your favorite terminal program, a text editor, and probably a version control system to keep track of changes to your content.
- Installation:

    apt-get install ansible

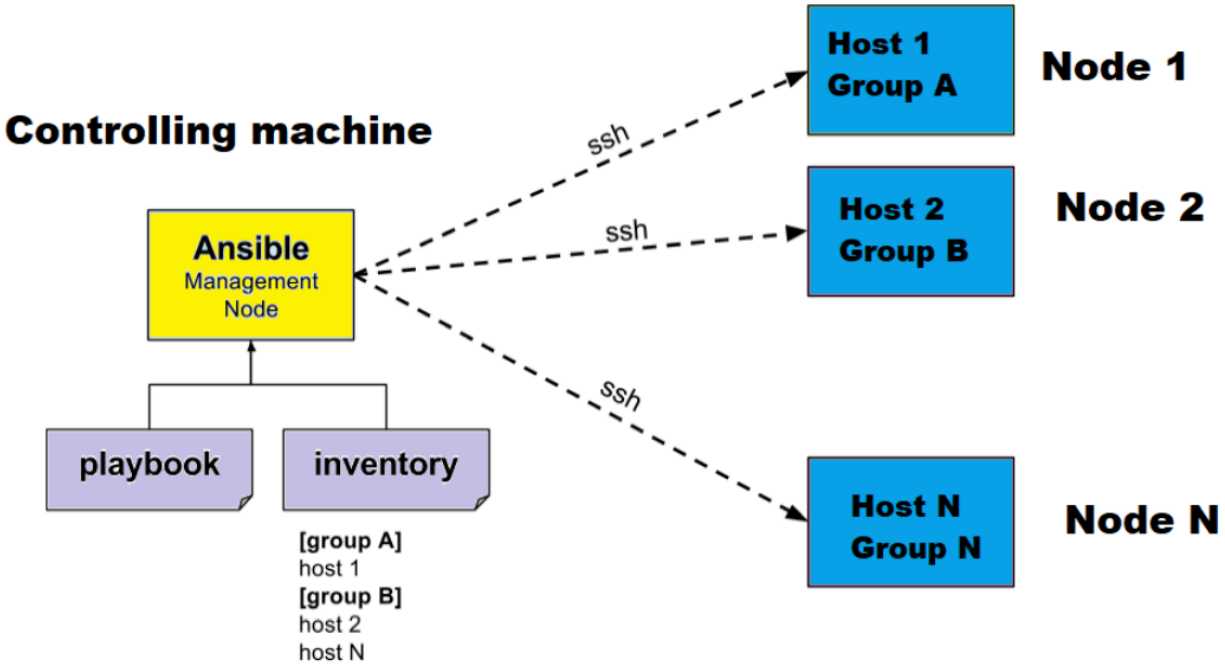
    yum install ansible

    dnf install ansible

    brew install ansible

    pip install ansible

ansible architecture



ansible component

- command line Tools:

  ansible, ansible-connection, ansible-doc, ansible-inventory, ansible-pull, ansible-vault, ansible-config, ansible-console, ansible-galaxy, ansible-playbook, ansible-test
- module
- Plugins
- filter
- Inventory (static,dynamic)
- playbook
- roles

Command line Tool

ansible

- AD-HOC command
- Usage Example:

```
ansible -i ”loaclhost,” localhost –m ping

ansible -i host.yml all –m module –a ”arg1=val1 arg2=val2”

ansible -i inventories/dev/host.yml vm02 -m setup -a \
                                    "filter=ansible_default_ipv4"
```

Command line Tool

ansible-connection

- Connection plugins allow Ansible to connect to the target hosts so it can execute tasks on them. Ansible ships with many connection plugins, but only one can be used per host at a time.
- By default, Ansible ships with several plugins. The most commonly used are the paramiko SSH, native ssh (just called ssh), and local connection types. All of these can be used in playbooks and with **/usr/bin/ansible** to decide how you want to talk to remote machines
- ansible-doc -t connection -l
- ansible-doc -t connection <plugin name>

Command line Tool

ansible-doc

- displays information on modules installed in Ansible libraries
- https://docs.ansible.com/ansible/2.4/ansible-doc.html

```
ansible-doc dnf -l
ansible-doc dnf -l | grep moduleName
ansible-doc dnf
ansible-doc dnf -s moduleName
```

Command line Tool

ansible-inventory

- Used to display or dump the configured inventory as Ansible sees it
- https://docs.ansible.com/ansible/latest/cli/ansible-inventory.html

```
ansible-inventory --list -i inventory.yml/ini
ansible-inventory --graph -i inventory.yml/ini
ansible-inventory --host localhost -i inventory.ini/yml
```

Command line Tool

ansible-pull

- pulls playbooks from a VCS repo and executes them for the localhost
- https://docs.ansible.com/ansible/latest/cli/ansible-pull.html
- is used to up a remote copy of ansible on each managed node, each set to run via cron and update playbook source via a source repository. This inverts the default push architecture of ansible into a pull architecture, which has near-limit less scaling potential.

Command line Tool

ansible-vault

- Encryption/Decryption utility for Ansible data files
- https://docs.ansible.com/ansible/latest/cli/ansible-vault.html

```
ansible-vault \
{create,decrypt,edit,view,encrypt,encrypt_string,rekey} \
        /path/to/vault.yml

ansible-playbook –i host.yml –l ”web” \
        webdeploy.yml –ask-vault-pass
```

Command line Tool

ansible-config

- view ansible configuration (read the details from default ansible.cfg).
- https://docs.ansible.com/ansible/latest/cli/ansible-config.html

```
ansible-config [-h] [--version] [-v] \
        {list,dump,view}  [-c,--config] /pathto/configfile.yml
```

Command line Tool

ansible-console

- Reason : Managing multiple machines sucks. No matter how much you automate there are always going to be edge cases where you'd like to perform the same command on multiple machines simultaneously.
- It’s connect host to ansible-console to do some module based operation
- https://docs.ansible.com/ansible/2.4/ansible-console.html
- Example:

  ansible-console -i inventories/dev/host.yml -l vm02

  ansible-console -i inventories/dev/host.yml

  ansible-console -i inventories/dev/host.yml –l vagrant

  After connect, you can see a prompt like below

  samitkumarpatel@vagrant (2)[f:5]$ module

Command line Tool

ansible-galaxy

- Ansible Galaxy refers to the Galaxy website, a free site for finding, downloading, and sharing community developed roles.
- It's not limit to Galaxy, we can manage galaxy role in any SCM tool, like github, bitbucket, etc..
- Use for create or install ansible role
- https://docs.ansible.com/ansible/latest/galaxy/user_guide.html

  ansible-galaxy init folderName

  ansible-galaxy install –r requirenment.yml –role-path=/path/to/folder

```
1 ---
2 - name: jenkins-jobloader
3   src: git+http://git.devops.apmoller.net/scm/ansible/jenkins-jobloader.git
```

Command line Tool

ansible-playbook

- An Ansible playbook is an organized unit of scripts that defines work for a server configuration managed by the automation tool AnsibleCan do a lot withit
- https://docs.ansible.com/ansible/latest/user_guide/playbooks_intro.html

  ansible-playbook -i inventories/dev/host.yml playbook.yml

```
2   - hosts: vm02
3     become: true
4     vars:
5       count: 0
6     tasks:
7       - name: ping
8         ping:
9       - name: apt update
10        apt:
11          upgrade: dist
12
13      - name: Install required system packages
```

Command line Tool

ansible-test

- https://docs.ansible.com/ansible/latest/dev_guide/testing_integration.html
- molecule - https://molecule.readthedocs.io/en/latest/
  - Drive- VM, docker,
  - Install – pip install molecule docker-py
  - molecule init role -r <<role_name>> -d <<driver>>
  - molecule test

module

- A **module** is a reusable, standalone script that **Ansible** runs on your behalf, either locally or remotely
- Modules (also referred to as “task plugins” or “library plugins”) are discrete units of code that can be used from the command line or in a playbook task. Ansible executes each module, usually on the remote target node, and collects return value
- We can write our own module based on our needs. You may write specialized modules in any language that can return JSON (Ruby, Python, bash, etc).
- https://docs.ansible.com/ansible/latest/modules/modules_by_category.html

You can execute modules from the command line:

```
ansible webservers -m service -a "name=httpd state=started"
ansible webservers -m ping
ansible webservers -m command -a "/sbin/reboot -t now"
```

```
- name: restart webserver
  service:
    name: httpd
    state: restarted
```

plugins

- Plugins are pieces of code that augment Ansible's core functionality.
- Ansible uses a plugin architecture to enable a rich, flexible and expandable feature set.
- Ansible ships with a number of handy plugins
- https://docs.ansible.com/ansible/latest/plugins/plugins.html

filter

- Filters in Ansible are from Jinja2, and are used for transforming data inside a template expression. ... Take into account that templating happens on the Ansible controller, not on the task's target host, so filters also execute on the controller as they manipulate local data.
- https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html

```
{{ some_variable | to_json }}
{{ some_variable | to_yaml }}
```

```
tasks:
  - shell: cat /some/path/to/file.json
    register: result

  - set_fact:
      myvar: "{{ result.stdout | from_json }}"
```

inventory

- The **Ansible inventory** file defines the hosts and groups of hosts upon which commands, modules, and tasks in a playbook operate.
- The file can be in one of many formats depending on your **Ansible** environment and plugins
- There are 2 types of inventory you can deal with – static and dynamic
- https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

Inventory - example

```
├── inventories
│   ├── dev
│   │   ├── group_vars
│   │   │   └── all
│   │   │       ├── vars.yml
│   │   │       └── vault.yml
│   │   ├── host.yml
│   │   └── host_vars
│   ├── pp
│   │   ├── group_vars
│   │   │   └── all
│   │   ├── host.yml
│   │   └── host_vars
│   └── test
│       ├── group_vars
│       │   └── all
│       ├── host.yml
│       └── host_vars
├── inventory
├── inventory.yml
├── inventory01.ini
├── inventory02.ini
├── inventory03.ini
├── playbook.yml
├── requirenment.txt
└── variable.yml
```

host.yml

```yaml
local:
  hosts:
    localhost:
      ansible_connection: local
vm01:
  hosts:
    192.168.33.10
vm02:
  hosts:
    192.168.33.11
vagrant:
  children:
    vm01:
    vm02:
all:
  children:
    local:
    vm01:
    vm02:
```

playbook

- An Ansible playbook is an organized unit of scripts that defines work for a server configuration managed by the automation tool Ansible.
- Ansible is a configuration management tool that automates the configuration of multiple servers by the use of Ansible playbooks
- https://docs.ansible.com/ansible/latest/user_guide/playbooks.html
- In playbook , you can use
  - Variable
  - Template (jinja2)
  - Condition
  - Loops
  - Block

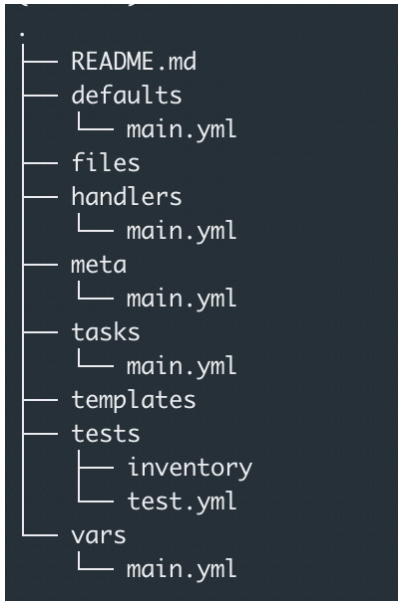
playbook -example

```yaml
- hosts: localhost
  connection: local
  tags: always
  roles:
    - role: buildset-resolve
      deploy_namespace: "{{ artifact.group_id.replace('.', '/') }}
      deploy_branch_prefix: "{{ artifact.version_prefix }}"
      when: artifact is defined

- hosts: localhost
  roles:
    - role: jenkins-job-list
      when:
        - jenkins is defined

- hosts: localhost
  roles:
    - role: usi-generate-files
      when: usi_collection is defined

- hosts:
    - weblogic
    - osb-primary
    - osb-secondary
  gather_facts: no
  pre_tasks:
    - name: setup
      setup:
      when: mw_deploy is defined
  roles:
    - role: middleware-ops-deploy
      when: mw_deploy is defined
```

```yaml
---
- hosts: all
  vars:
    deploy: A
  tasks:
    - name: print expected variable
      with_items: "{{ defined_groups }}"
      debug:
        msg: "{{  item }}"
      when:
      - ( item.deploy_tag | intersect(deploy) ) or deploy is not defined
```

roles

- Roles provide a framework for fully independent, or interdependent collections of variables, tasks, files, templates, and modules.
- In Ansible, the role is the primary mechanism for breaking a playbook into multiple files.
- Note- Roles are not playbooks



demo:1

- Write a playbook to:
  - Provision a machine with following software
    - docker
    - docker-compose
    - enable docker-swarm
  - Requirenment is: Install Jenkins

demo:2

- Convert the Jenkins install playbook to a role

ansible vs terraform

- DEMO1
  - Create ansible playbook to create infrastructure(vnt,public ip,vm)
  - Install software with ansible(docker,docker-swarm)
  - Install Jenkins
- DEMO2
  - Create terraform module to create infrastructre
  - Install software with ansible
  - Install Jenkins